DALock: Password Distribution Aware Throttling

Jeremiah Blocki
Purdue University

Wuwei Zhang
Purdue University

Abstract

Large-scale online password guessing attacks are wide-spread and continuously qualified as one of the top cyber-security risks. The common method for mitigating the risk of online cracking is to lock out the user after a fixed number (k) of consecutive incorrect login attempts within a fixed period of time (e.g., 24 hours). Selecting the value of k induces a classic security-usability tradeoff. When k is too large a hacker can (quickly) break into a significant fraction of user accounts, but when k is too low we will start to annoy honest users by locking them out after a few mistakes. Motivated by the observation that honest user mistakes typically look quite different than the password guesses of an online attacker, we introduce the notion of a *password distribution aware* lockout mechanism to reduce user annoyance while minimizing user risk. As the name suggests, our system is designed to be aware of the frequency and popularity of the password used for login attacks while standard throttling mechanisms (e.g., k-strikes) are oblivious to the password distribution. In particular, we maintain an "hit count" for each user which is based on (estimates of) the cumulative probability of *all* login attempts for that particular account. A user will only be locked out when this hit count is too high. To minimize user risk we use a differentially private CountSketch to estimate the frequency of each password and to update the "hit count" after an incorrect login attempt. To empirically evaluate our new lockout policy we generate a synthetic dataset to model honest user logins in the presence of an online attacker. The result of our analysis on this synthetic dataset strongly support our hypothesis that distribution aware lockout mechanisms can simultaneously reduce both user annoyance *and* risk.

Who Are You?! Adventures in Authentication (WAY) 2019.
August 11, 2019, Santa Clara, CA, USA.

1 Introduction

1.1 Background

Dictionary attack is one of the easiest yet effective cyber-security attacks. Within a few attempts of popular passwords, adversaries are able to compromise a significant amount of accounts [13]. Unfortunately, a well-ordered password dictionary can be effortlessly constructed from two disappointing facts: weak passwords reused across websites [20] and data breaches [21].

Classical login throttling mechanism defends against dictionary attacks by limiting maximum attempts K within a fixed time window; However, unwanted throttling can occur frequently due to users' honest mistakes [4]. To increase usability, fault-tolerant password authentication systems were purposed [4] [5] and deployed [14]. When configured properly, they offer utility improvement without compromising security. Unfortunately, these systems often require storing personal information to determine predefined typos. Another fly in the ointment is not able to tolerate mistakes other than predefined ones.

To the best of our knowledge, there is no existing password authentication system offers high usability without compromising security by not storing individual information such as possible typos.

1.2 Contributions

To answer the call, we purpose a novel password distribution aware throttling mechanism: DALock. DALock works by keeping track of a "hit count" ψ and consecutive incorrect attempts k for each user and locking the user account when any of these two values gets too high. DALock updates ψ based on the popularity of the attempted passwords. Therefore Adversaries are discouraged to try popular passwords because ψ can get skyrocketed. Speaking of usability, Users are less

likely to trigger a properly configured DALock because their honest mistakes tend to be infrequent or even non-existing passwords.

To run DALock, one needs to be able to efficiently estimate frequencies of passwords without disclosing private information. Differential private Count-Sketch is a natural choice for the task. In this work, we discovered the optimal usage of Count-Sketch for storing frequencies of passwords. We empirically showed that Count-Median-Sketch is the best among Count-Mean-Sketch, Count-Median-Sketch, and Count-Min-Sketch for storing password distribution. In addition, we discovered the optimal parameter setting of Count-Sketches.

We prove that any rational adversary performs large-scale online guessing attacks on DALock is challenged by a classic NP Complete problem called "Knapsacks". We further proved and empirically verified that any rational online attacker are not able to compromise more users on DALock than tradition throttling mechanism. We also empirically show that users are benefited from DALock without compromising security when DALock is properly configured.

In this work, we focus on online dictionary attack scene. We assume that adversaries perform large-scale untargeted online attacks to maximize their economic gain. i.e., the adversaries try to break as many accounts as possible. We assume adversaries have efficient computational power, for example, bitcoin mining machines. For security practice, we assume the adversaries also have precise knowledge of the distribution of the passwords and deployment details of DALock.

2 Background and Related Works

Traditional Throttling Mechanism And Its Variants. Traditional throttling mechanism often forces users to change their password if too many incorrect attempts are made within a short period of time. This offers reliable defense against dictionary attacks [2]; however, users are also penalized by it due to their frequent mistakes such as typos. To make the mechanism more fault-tolerant, multiple mechanisms were purposed. One alternative is to promote Automated Turing Test (ATT) such as CAPTCHA [17] instead of locking the account. Unfortunately, studies show that users hate ATT [22] while adversaries can circumvent it [8]. Typo-tolerant approaches [5] [4] allow users login their accounts when typos are entered. Despite the fact that such approaches offer tremendous usability without sacrificing security, those system have to store personal information about password typos to facilitate fault tolerance.

Human Generated Passwords Despite many efforts were made from academia and industries to educate users to choose strong passwords [16], the distribution of passwords remains pessimistic [10]. Literatures [18] [19] showed that human generated passwords generally follow Zipf's distribution. Dictionary attackers are benefited from such distribution because they can compromise significant amount of accounts by attempting popular ones. Naor et al [12] argue that popular passwords should be banned without publishing (to prevent adversaries acquire the distribution of passwords) so that no single password can compromise many accounts. There are two major drawbacks. Firstly, users tend to choose weak passwords, banning known popular passwords just make them choose other weak passwords. Secondly, despite the fact that not publishing the distribution is a good practice, one can't assume the adversaries don't have strong knowledge of it. Schechter et. al [15] suggest the fix is to "force" the password distribution converges to uniform distribution by upper bounding the number of repeated passwords. This is a realistic password policy, but not a replacement for password authentication system.

Password Typos Chatterjee et al. [4] empirically measured frequent password typos. In their experiment, volunteers were asked to enter randomly sampled passwords from RockYou [11] passwords leakage. Based on their result, there were 4,364 incorrect submissions across 97,632 valid submissions (4.5%). In addition, 5.5% of 81,595 unique passwords were mistyped at least once. Over 42% of the participants made at least one mistake. Five common types(21.5%) discovered are: 1) switching all symbol cases(10.9%), 2) adding extra letters at the end(4.6%), 3) switching first symbol case(4.5%) , 4) adding extra letters at the front(1.3%), and 5) missing "shift" for last symbol(0.2%). They also purposed an approximately optimal construction for typo-tolerant password checker based on experiment results. The password checker allows users to login their accounts when common typos are attempted. They show that 20% of the volunteers could have login 1 minute earlier with the help of password checker.

Defense Against Online Dictionary Attacks. To enhance the usability and security of classic throttling mechanism, many "add-on" methodologies were purposed. By utilizing carefully engineered features, machine learning [9] approach is able to identify suspicious login attempts. To increase the financial cost of adversaries, Golla [7] purposed a fee-based password verification system. The system charges real currency for each login attempts and makes refund if login is successful. As a result, online attacks can be costly due to a huge multiplicative factor: number of accounts. Two Factor Authentication(2FA) [1] is a popular methodology to stop dictionary attacks. 2FA challenges attackers by posing real time challenges such as verification code. StopGuessing [23], a concurrent work with similar idea to us, is an IP-based throttling mechanism. An IP address is banned if too many popular passwords were attempted.

3 The DALock Mechanism

In this section, we present the DALock mechanism, discuss how DALock might be implemented and the strategies that

an attacker might use when DALock is deployed.

3.1 DALock

We formally introduce our main algorithm, DALock, in this section. We start by defining and describing the general framework of DALock. Following by proving why it is secure and user-friendly.

We first formally define classic throttling mechanism. Traditional Throttling mechanism maintains a counter k for each user u. Counter k stores consecutive incorrect login attempts in history. Typically, k is reset to 0 whenever u logins successfully. Some systems also reset k to 0 after a certain amount of time or maintain separate counters for different IP addresses. Mathematically speaking, one can define classic throttling mechanism as follows

Definition 1 (K-Secure Throttling Mechanism) *Given integer $K \geq 1$, K-Secure Throttling Mechanism $\mathcal{M}$ prevents login if $k \geq K$.*

DALock maintains extra "hit count" ψ corresponding to the total population density of attempted passwords. For example, if three incorrect passwords "aaa"(3%), "bbb"(1.7%), and "lccc"(0.8%) are attempted, then ψ is set to 0.055.

Definition 2 ((Ψ,K)-Secure Throttling Mechanism) *Given $\Psi > 0$ and integer $k \geq 1$, (Ψ,K)-secure throttling mechanism $\mathcal{M}$ prevents login if*

$$k \geq K \text{ or } \psi \geq \Psi$$

We demonstrate the login flow in **Algorithm** 1. The pseudo code is self-explanatory, therefore we omit the detail description of it.

Algorithm 1 DALock: Novel Password Distribution Aware Throttling Mechanism

Input: Username u and password p

```
1:  function LOGIN(u, p)
2:      if ψ ≥ Ψ or k ≥ K then
3:          Reject Login
4:      end if
5:      if p == u_p then
6:          Reset k and ψ
7:          Grant Access
8:      else
9:          ψ ← ψ + popularity(p)
10:         k ← k + 1
11:         Deny Access
12:     end if
13: end function
```

3.2 Implementing DALock

To efficiently run DALock, one needs an efficient data structure to accurately, privately, and securely estimate the population density of passwords. We adopt Differential Private Count-Median-Sketch to store password distribution as it meets all the expectations. Due to space limitation, we omit the description of applying differential privacy to focus on discussion Count Sketch.

Definition 3 (Count Sketch [6] [3]) *A Count sketch with volume $V = w \cdot d$ is represented by a two-dimensional array with width w and depth d. Additionally, it contains d + 1 hash functions*

$$h_1 \cdots h_d : \{\sigma^*\} \to \{1 \cdots w\}$$
$$h_\pm : \{\sigma^*\} \to \{1, -1\}$$

are chosen uniformly at random from a pairwise-independent family. Finally an integer T is maintained to record the total frequency.

A typical Count Sketch involves the following operations: **Add(p):** Given input password word $p = \sigma^*$, CS updates the table as follows:

$$\forall i \in d, CS[i, h_i(p)] \leftarrow CS[i, h_i(p)] + h_\pm(p)$$
$$T \leftarrow T + 1$$

Estimate(p): Given input password word $p = \sigma^*$, CS estimates its popularity based on predefined mechanism.

In this work, we consider three popular estimation functions: Median, Mean, and Min. Median approach estimate the frequency of p by $Median_{\forall i \in d}(\, CS[i, h_i(p)] \cdot h_\pm(p))$. Similarly, the other two approach use the mean and min of d values as estimation.

We discovered that Count-Median-Sketch outperforms Count-Mean-Sketch and Count-Min-Sketch for storing password distribution. In addition, we discovered that setting d to 1 yields the lowest l_1 error for all CS. Subject to the space limitation, we only present the high level proof for Count-Median Sketch.

Theorem 1 (Optimal Parameter Choice of Count Sketch) *Given volume size $V = d \cdot w$, the optimal choice of Count Sketch, Count Mean Sketch, and Count Min Sketch is to set d = 1 when passwords follow Zipf's Distribution.*

Intuitive Proof: Due to the nature of Zipf's distribution, the frequency count of any index CS[i,j] of a count median sketch is dominated by the most popular password hashed into that index. With larger d, an infrequent password p is more likely to collide with significantly more popular passwords on every row. Taking the median of d rows is essentially taking the median of d different popular passwords. Therefore, infrequent passwords are overwhelmingly overestimated with large d.

3.3 Attacker's Strategies

In order to launch optimal dictionary attacks, $\mathcal{A}$ needs to find a subset of passwords that maximize its chance of success **without** triggering DALock. We proved this is computational challenging. In fact, it is NP hard to find the optimal guessing arrangement.

Hardness of Password Cracking Perform optimal dictionary attacks on(Ψ, K)-Secure throttling mechanism is NP hard.

To maximized its chance of success, $\mathcal{A}$ has to select at most $K-1$ passwords from the dictionary s.t. the sum of popularity is maximized **and** not exceeding Ψ. This can be reduced to solving a well-known NP hard problem: Knapsacks.

Admittedly, $\mathcal{A}$ can solve Knapsacks efficiently with close approximation because Knapsacks $\in$ FPTAS. Fortunately, users' unpredictable mistakes make the attack harder. $\mathcal{A}$ is forced to make a decision about how much "mistake budget"ψ' to be reserved for users' potential mistakes.

Creating false popular passwords is another way to exploit DALock. In this circumstance, $\mathcal{A}$ needs to sign up sufficiently many accounts with infrequent passwords to decrease the popularity of true ones. Despite the fact that it can circumvent the restriction imposed by Ψ, $\mathcal{A}$ is still subject to counter K. Therefore $\mathcal{A}$ does not gain any advantage by disturbing distribution.

4 Experiments

In this section, we empirically demonstrate the advantages of DALock. Firstly, we demonstrate DALock offers higher utility without sacrificing security by short term experiments. Secondly, we demonstrate that DALock can also effectively defend against powerful adversaries when attack is carried in long time span.

4.1 Utility of DALock

In this section, we empirically show that DALock significantly reduces unwanted lockouts without compromising security.

Experiment Settings For each run of this experiment, we simulate 100,000 users and create their passwords based on RockYou [11], a leaked data set contains 32 millions passwords. We independently measure security and utility in short time window. Count-Median-Sketch used in this experiment has Volume V = 2,000,000 with optimal parameters setting, i.e. d = 1 and $w = V$. We verify the performance of DALock with the following combinatorial settings of K and Ψ.

- K: 3,5,10,20,50,100
- Ψ: $2^0, 2^{-2}, 2^{-4}, 2^{-6}, 2^{-8}, 2^{-10}$

Security Measurement We quantify security of the system by counting how many accounts $\mathcal{A}$ can compromise without threshold reset. In this setting, $\mathcal{A}$ attempts passwords on every account based on the popularity until those accounts are locked or cracked. **Figure** 1 illustrates the result of this measurement. The darkness of each pixel represents the percentage of accounts get compromised. The X-axis represents the value of Ψ in log scale with base 2. The Y-axis represents k. For example, the top right pixel represents that $\mathcal{A}$ is able to compromise 4.6% of users' accounts when k = 100 and $\Psi = 2^0 = 1$.

Utility Measurement Similarly, utility is quantified by unwanted lockouts. We simulate realistic short term scenarios as follows. Each user u keeps submitting passwords (with a chance of making mistake) until u successfully login or triggers DALock (with consecutive errors). Users' mistakes are simulated based on Amazon Mturk Experiments [4].

Figure 2 shows the probabilities of getting unwanted lockout under various configurations of DALock. Similar to **Figure** 1, X and Y axes represent Ψ and K correspondingly. The darkness of the pixel indicates the frequencies of unwanted lockouts in percentage.

Discussion By comparing the result of two heatmaps, one can observe that:

- Ψ has tiny or no impact on unwanted lockouts.
- Ψ and K both impact the chance success for $\mathcal{A}$
- Ψ sharply reduce the chance of being compromised.

Given the fact that roughly 1% of the users are using the most popular password. Setting $\Psi < 1\%$ $2^{-6.6}$ makes it more powerful than K. **Figure** 1 also confirms there is a significant drop between $\Psi = 2^{-6}$ and 2^{-8}.

DALock improves the utility dramatically by increase K without compromising security when Ψ is sufficiently small.

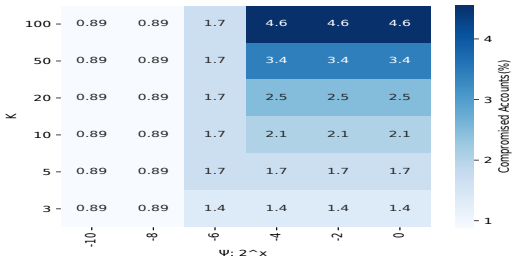

Figure 1: Short Term Security Measurement

4.2 Long Term Attack Experiment

In reality, dictionary attacks can be carried over long time span to prevent triggering throttling. In this experiment, we further demonstrate DALock can effectively defend against powerful and realistic adversaries in long term. Two types of adversaries are considered in our works:

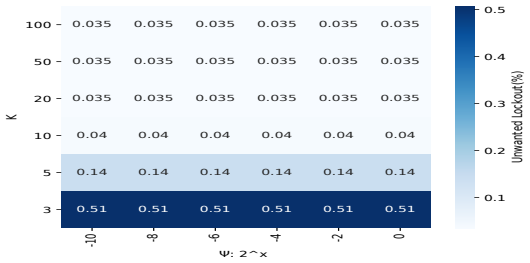

Figure 2: Short Term Utility Measurement

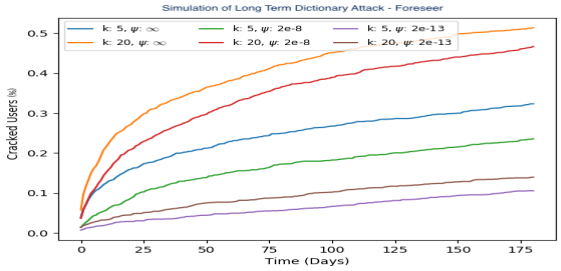

Figure 3: Foreseer

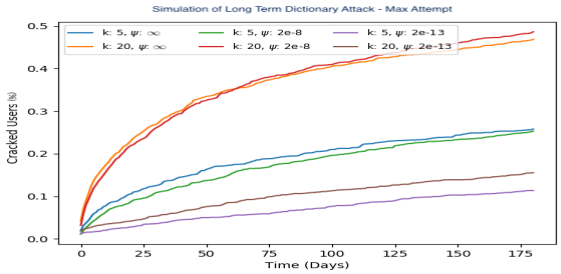

Figure 4: Max Attempts

- **Foreseer**: $\mathcal{A}_F$(Foreseer) is able to foresee users' activities include whether they will type in their password correctly or not. $\mathcal{A}_F$ never triggers DALock because of this advantage.
- **Max Attempts**: $\mathcal{A}_M$ is realistic yet powerful. $\mathcal{A}_M$ is able to observe accounts activities such as threshold resets to facilitate its attack. $\mathcal{A}_M$ attempts up to K-1 passwords after each threshold reset.

Experiment Setting: We simulate the attack with 100,000 accounts over time span of 180 days. Users' login activities are simulated based on poisson arriving process. Their passwords [11] and typos are simulated as described in aforementioned utility experiment. The Count-Median-Sketch used in DALock has volume $V = 200,000$ and $d = 1$. Both $\mathcal{A}_F$ and $\mathcal{A}_M$ makes sub-optimal guessing arrangement based on greedy algorithm.

We demonstrate the results under six configurations of K and *Psi*: (5,∞), (5,2e-8), (5,2e-13), (50,∞), (50,2e-8), and (50,2e-13). Notice that when $\Psi = \infty$, the mechanism is just traditional throttling methods.

Foreseer: **Figure** 3 demonstrates the progress of attack over 180 days for $\mathcal{A}_F$. This is essentially the maximum amount of accounts one can crack **without** triggering DALock. The X-axis corresponds to time span by days and Y-axis represents the percentage of accounts get compromised.

Max Attempt: **Figure** 4 demonstrates the progress of attack over 180 days for $\mathcal{A}_F$. Similarly to **Figure** 3, X and Y of **Figure** 4 axes stand for time and amount of compromised accounts in percentage.

Discussion: One can conclude that DALock is significantly safer compare to traditional throttling mechanism. Despite a powerful adversary like $\mathcal{A}_F$, the chance of success is reduced sharply as Ψ drops. In addition, when Ψ is set to sufficient small value. One can enlarge k to increase users' utility.

5 Limitation and Discussion

Comparison to Other Defense StopGuessing [23], A concurrent work, also uses popularity of passwords for throttling; however we argue that our mechanism is different and better in several ways. Firstly, StopGuessing blocks IP address that attempts popular passwords in lieu of freezing users' accounts. This can be problematic because there are multiple ways to circumvent the restrictions. For example, using botnet to launch distributed attacks. Secondly, we argue that our methodology of storing password distribution is superior because of differential privacy. Thirdly, StopGuessing increases threshold when successful login are made. This can be problematic for users with weak passwords and login frequently.

Limitations DALock assumes that adversaries perform rational online dictionary attacks. Targeted attacks are beyond the scope of defense. Another limitation is not be able to protect users who use the most frequent passwords. Because 1 attempt is sufficent to compromise their accounts. Fortunately, DALock can be integrated with other systems such as 2FA to mitigate those issues. Final limitation comes from differential privacy, one cannot lively publish password distribution due to privacy budget restriction. This can be solved by publishing the distribution less frequently or using a known password distribution.

Conclusion

In this work, we purposed a novel throttling mechanism DALock that utilize privately collected passwords distribution to defend against online dictionary attacks. DALock provides utility improvement without sacrificing security. Finally, we discovered that the optimal usage of Count Sketches for stor-

ing passwords.

References

[1] Fadi Aloul, Syed Zahidi, and Wassim El-Hajj. Two factor authentication using mobile phones. In *2009 IEEE/ACS International Conference on Computer Systems and Applications*, pages 641–644. IEEE, 2009.

[2] Mansour Alsaleh, Mohammad Mannan, and Paul C Van Oorschot. Revisiting defenses against large-scale online password guessing attacks. *IEEE Transactions on dependable and secure computing*, 9(1):128–141, 2012.

[3] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

[4] Rahul Chatterjee, Anish Athayle, Devdatta Akhawe, Ari Juels, and Thomas Ristenpart. password typos and how to correct them securely. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 799–818. IEEE, 2016.

[5] Rahul Chatterjee, Joanne Woodage, Yuval Pnueli, Anusha Chowdhury, and Thomas Ristenpart. The typtop system: Personalized typo-tolerant password checking. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 329–346. ACM, 2017.

[6] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[7] Maximilian Golla, Daniel V Bailey, and Markus Dürmuth. "i want my money back!" limiting online password-guessing financially. In *SOUPS*, 2017.

[8] Philippe Golle. Machine learning attacks against the asirra captcha. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 535–542. ACM, 2008.

[9] Sakshi Jain. Who are you? a statistical approach to measuring user authenticity. In *The Network and Distributed System Security Symposium (NDSS) 2016*. Internet Society, 2016.

[10] David Malone and Kevin Maher. Investigating the distribution of password choices. In *Proceedings of the 21st international conference on World Wide Web*, pages 301–310. ACM, 2012.

[11] Daniel Miessler. Seclists.

[12] Moni Naor, Benny Pinkas, and Eyal Ronen. How to (not) share a password: Privacy preserving protocols for finding heavy hitters with adversarial behavior. *IACR Cryptology ePrint Archive*, 2018:3, 2018.

[13] Thu Pham. Stop the pwnage: 81% of hacking incidents used stolen or weak passwords.

[14] Emil Protalinski. Facebook passwords are not case sensitive.

[15] Stuart Schechter, Cormac Herley, and Michael Mitzenmacher. Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks. In *Proceedings of the 5th USENIX conference on Hot topics in security*, pages 1–8. USENIX Association, 2010.

[16] Eugene H Spafford. Opus: Preventing weak password choices. *Computers & Security*, 11(3):273–278, 1992.

[17] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. Captcha: Using hard ai problems for security. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 294–311. Springer, 2003.

[18] Ding Wang, Haibo Cheng, Ping Wang, Xinyi Huang, and Gaopeng Jian. Zipf's law in passwords. *IEEE Transactions on Information Forensics and Security*, 12(11):2776–2791, 2017.

[19] Ding Wang and Ping Wang. On the implications of zipf's law in passwords. In *European Symposium on Research in Computer Security*, pages 111–131. Springer, 2016.

[20] Rick Wash, Emilee Rader, Ruthie Berman, and Zac Wellmer. Understanding password choices: How frequently entered passwords are re-used across websites. In *Twelfth Symposium on Usable Privacy and Security ({SOUPS} 2016)*, pages 175–188, 2016.

[21] Zack Whittaker. Facebook now says its password leak affected 'millions' of nstagram users.

[22] Jeff Yan and Ahmad Salah El Ahmad. Usability of captchas or usability issues in captcha design. In *Proceedings of the 4th symposium on Usable privacy and security*, pages 44–52. ACM, 2008.

[23] Stuart Schechter Yuan Tian, Cormac Herley. Stopguessing: Using guessed passwords to thwart online guessing. In *4th IEEE European Symposium on Security and Privacy*. IEEE, 2019.